Niko Martikainen

Keskitetty valvontaratkaisu järjestelmä-integraatioille moniasiakasympäristössä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikka

Insinöörityö

27.11.2017

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Niko Martikainen Keskitetty valvontaratkaisu järjestelmäintegraatioille moni-asiakasympäristössä 31 sivua + 1 liite 27.11.2017
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikan ko.
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	Systems Developer Juuso Sormunen Yliopettaja Auvo Häkkinen

Insinöörityön tavoitteena oli tehostaa järjestelmäintegraatioiden valvontaa moniasiakasympäristössä ottamalla käyttöön keskitetty valvontaratkaisu. Valvonta-alustaksi valittiin Zabbix-monitorointityökalu. Valvontaratkaisulla pyrittiin parantamaan toimittajan reagointiaikaa integraatio-ongelmissa ja mahdollistamaan parempi virheiden ennaltaehkäisy sekä tuomaan lisäarvoa ongelmien selvittämiseen.

Työssä konfiguroitiin Zabbix valvomaan toimittajan iPaaS-ratkaisun (Integration Platform as a Service) verkkoinfrastruktuuria, integraatioita sekä asiakasympäristöjen integraatioratkaisuja. Työn teoriaosuuksissa pyrittiin antamaan tarpeellinen kokonaiskuva integraatioista ja Zabbix-valvontaohjelmistosta sekä pohdittiin valvontaratkaisun onnistumista ja jatkokehitystä.

Projektissa saavutettiin sille asetetut tavoitteet. Zabbix toimi hyödyllisenä monitorointityökaluna IT-infrastruktuurin lisäksi myös integraatiolle ominaisten valvontakomponenttien valvontaan. Se mahdollisti joustavasti asetettavat valvontamääritykset, jossa tietoturvatarpeet ja erilaiset verkko- ja integraatioratkaisut pystyttiin huomioimaan.

Avainsanat	Järjestelmäintegraatio, EAI, Zabbix

Author(s) Title Number of Pages Date	Niko Martikainen Centralized Monitoring Solution for Systems Integrations in a Multi Customer Environment 31 pages + 1 appendix 27.11.2017
Degree	Bachelor of Engineering
Degree Programme	Information and Communications Technology
Specialisation option	Software Engineering
Instructor(s)	Juuso Sormunen, Systems Developer Auvo Häkkinen, Principal Lecturer

The aim of this thesis was to improve the current monitoring solutions for systems integrations by introducing a new centralized monitoring system. Zabbix monitoring software was chosen as the new solution to improve the provider's ability to response to integration problems and to enable proactive approach to error detection.

Zabbix was configured to monitor the provider's iPaaS-solution's (Integration Platform as a Service) network infrastructure, systems integrations as well as integration solutions on customers' networks. The two concrete monitoring targets for integrations introduced in the thesis are messaging queue monitoring and silence monitoring.

The thesis gives an overview of systems integrations and the Zabbix monitoring tool. The further development ideas of the solution are also presented.

As a result of the project, the monitoring system was found to be a valuable solution for monitoring IT-infrastructure and integration specific components. It enabled flexible customization in cases where different security, network and integration needs could be taken into account.

Keywords	EAI, Systems integration, Zabbix, Monitoring

Sisällys

Lyhenteet ja termit

B2Bi	*Business to Business Integration.* Yritysten välinen integraatio.
EAI	*Enterprise Application Integration.* Yritysjärjestelmien integraatio.
ESB	*Enterprise Service Bus.* Yrityksen palveluväylä, jolla mahdollistetaan erilaisilla arkkitehtuureilla toteutettujen järjestelmien liittäminen toisiinsa.
HTTP	*Hypertext Transfer Protocol.* Hypertekstin siirtoprotokolla.
JMS	*Java Message Service.* Viestinvälitysmalli, joka käyttää hyväkseen jonoja.
JMX	Java Management Extensions. Java-teknologia, jolla mahdollistetaan sovelluksen etävalvonta ja -hallinta.
SLA	*Service Level Agreement* eli palvelutasosopimus. Asiakkaan ja palvelun-tarjoajan välinen sopimus, jossa määritellään palvelulle tietyt vaatimustasot.
SOA	*Service Oriented Architecture* eli palvelukeskeinen arkkitehtuuri. Arkkitehtuuritason suunnittelutapa.
XML	*Extensible Markup Language*. Yleisesti sanomaformaattina käytettävä metakieli.
XSLT	*Extensible Stylesheet Language Transformations.* XML-tiedostojen muunnoksiin tarkoitettu kieli.

1. Johdanto

Järjestelmäintegraatioilla mahdollistetaan yritysten eri järjestelmien kommunikointi keskenään. Liiketoiminnan kannalta välttämättömän tiedon hyödyntäminen tehokkaasti kasvattaa liiketoiminta-arvoa, kun manuaaliset ja virhealttiit toimenpiteet saadaan korvattua automatisoiduilla ratkaisuilla. Valvonnan rooli on varmistaa, että nämä kriittiset tiedonjyväset löytävät oikeaan paikkaan, oikeaan aikaan ja oikealla tavalla.

Työn tavoitteena on löytää keskitetty, automatisoitu ja reaaliaikainen valvontaratkaisu integraatioiden valvontaan. Useiden kohteiden valvonta nopeutuu ja helpottuu keskitetyillä ratkaisuilla, kun kaikki tieto löytyy yhdestä paikasta. Reaaliaikainen valvonta mahdollistaa toimenpiteiden tekemisen ennen kuin ongelma ehtii eskaloitua ja yrityksen liiketoiminta vaarantua. Valvonnan automatisointi taas vähentää työlästä ja virhealtista käsityötä.

Insinöörityö toteutettiin kehitystyönä Digia Finland Oy:n integraatioiden jatkuvien palveluiden ylläpidosta ja pienkehityksestä vastaavalle noin 15 hengen tiimille, jolla on useiden eri asiakkaiden integraatioympäristöjä valvottavanaan. Työn tarkoituksena on parantaa olemassa olevaa valvontaa ja laajentaa asiakaskohtainen valvonta keskitetyksi, usean eri asiakkaan integraatioratkaisuiden valvonnaksi.

Työssä käydään ensin läpi järjestelmäintegraatioiden keskeiset asiat, kuten miksi ja miten integraatioita ylipäätänsä tehdään. Tämän jälkeen kerrotaan valvonnan nykytilasta ja tavoitteesta, johon uudella valvontaratkaisulla pyritään pääsemään. Syyt valvonnan lisäämiselle sekä onnistumiskriteereiden määrittely kahdelle esimerkkivalvonnalle tulee tämän jälkeen. Zabbix-luvussa kerrotaan monitorointityökalu Zabbixin työn kannalta oleellisimmista ominaisuuksista ja toiminnallisuuksista. Valvonnan suunnittelua ja kehittämistä käsitellään tämän jälkeen. Lopuksi arvioidaan työn onnistumista yleisesti ja asetettujen onnistumiskriteereiden avulla sekä pohditaan valvonnan tulevaisuutta.

2. Järjestelmäintegraatio

Ennen organisaatioilla oli käytössään yksi tai useampi järjestelmä, joiden välillä ei ollut minkäänlaista yhteyttä. Näiden monoliittisten järjestelmien keskinäinen tiedonsiirto hoidettiin manuaalisesti, joka oli sekä työlästä että hidasta. Nykyään automatisoinnin avulla näistä eri tarkoituksiin luoduista palasista rakentuu yhteistoiminnallisia tietojärjestelmäkokonaisuuksia, jotka toimivat kuin yksi kattava järjestelmä. Tätä järjestelmien yhdistämistä kutsutaan järjestelmäintegraatioksi. [1, s. 15-16.]

Yrityksillä on käytössään vielä paljon legacy-sovelluksia, joita ei ole suunniteltu integraatioita silmällä pitäen ja jotka usein eroavat merkittävästi uudemmista sovelluksista arkkitehtuureiltaan ja teknologioiltaan. Yritysten tulee voida ottaa käyttöön uusia sovelluksia liiketoiminnan tarpeiden mukaan. Kaiken kattavan tietojärjestelmän kehittäminen on käytännössä mahdotonta yrityksen tarpeiden kasvaessa ja vaihdellessa. Myös vanhojen sovellusten vaihtaminen uusiksi yhteensopiviksi järjestelmän komponenteiksi on usein liian kallista ja vie aikaa. Järjestelmäintegraatiot mahdollistavat kriittisen tiedon saatavuuden järjestelmien välillä lähes aina kustannustehokkaammin. [2, s. 25-30.]

Järjestelmäintegraatio vastaa siis ongelmaan, kuinka toisistaan eriävät järjestelmät voivat kommunikoida toistensa kanssa tuntematta niitä tai tietämättä niiden rajoituksista. Sen ei tule ottaa kantaa sovellusten sisäiseen kommunikointiin eikä keskenään samanlaisten sovellusten väliseen kommunikointiin. [3, s. 23.] Se ei myöskään viittaa mihinkään yksittäiseen teknologiaan, vaan tiettyihin vakiintuneisiin käytäntöihin ja malleihin. Se on tapa tehostaa yrityksen eri järjestelmien välistä kommunikointia, jotka eivät olisi muutoin yhteensopivia. Tiedon siirto itsessään ei ole vielä järjestelmäintegraatioita, vaan tieto pitää pystyä muuttamaan järjestelmien ymmärtämään muotoon. Se pitää sisällään myös sanomaliikenteen kontrollointia, valvontaa ja raportointia. [1, s. 15-16, 48.]

Liiketoiminnan kannalta tavoite on maksimoida jokaisen yksittäisen sovelluksen ja koko tietojärjestelmän hyötyjä. Tiedon yhtenäinen ja tehokas saatavuus lisää kilpailuetua. [2, s. 27-28.] Järjestelmäintegraatioilla, kuten muillakin liiketoimintaa tukevilla sovelluksilla, haetaan tehokkuutta. Toimintojen automatisointi ja keskittäminen vähentävät kallista ja virhealtista käsityötä. Kun tieto jaetaan järjestelmien kesken, tietoa voidaan laittaa yhteen järjestelmään ja integraation avulla myös muut tietoa tarvitsevat järjestelmät pääsevät tietoon käsiksi. Järjestelmäintegraatioiden mahdollistamalla joustavuudella saa-

daan pidennettyä ohjelmistojen elinikää, sillä kokonaisratkaisua ei tarvitse uusia kokonaan, vaan voidaan lisätä halutun ominaisuuden omaava komponentti kokonaisratkaisuun. Integraatiot lisäävät joustavuutta myös mahdollistamalla suuret organisaatiomuutokset ja vähentämällä valmistajariippuvuutta. [1, s. 22-30.] Järjestelmien loppukäyttäjille integraatiot toimivat parhaimmillaan näkymättömästi [1, s. 63] ja reaaliaikaisesti [2, s. 26].

Järjestelmäintegraation käsite vaihtelee lähteestä riippuen. Järjestelmäintegraatioista käytetään usein nimitystä EAI (Enterprise Application Integration) [4], vaikka toisinaan sillä tarkoitetaan järjestelmäintegraatioiden teknistä ratkaisua [5] ja toisinaan sitä pidetään yrityksen sisäisenä järjestelmäintegraationa. Suomenkielisessä kirjallisuudessa käytetään myös usein nimitystä sovellusintegraatio. [1, s. 16, 33] Nykypäivän liiketoiminnan tarpeet luovat tarpeen myös yritysten välisille integraatioille. Ulkoisiin järjestelmiin integroitaessa puhutaan B2Bi:stä (Business to Business Integration) tai B2Bia:sta (Business to Business Application Integration). [1, s. 34; 2, s. 51-52.]

2.1 Tiedonsiirtomenetelmät

Integraatioiden mahdollistamiseksi tarvitaan menetelmät siirtää informaatiota järjestelmien välillä. Rajapinnat mahdollistavat järjestelmien keskustelun keskenään tarjoamalla tavat syöttää ja hakea tietoa järjestelmistä. [1, s. 49.] Järjestelmäintegraatiokontekstissa rajapintojen voidaan ajatella olevan tietynlaisia portteja järjestelmään ja sieltä ulos.

Järjestelmien välinen kommunikointi voi olla joko synkronista tai asynkronista. Synkroninen kommunikaatio on reaaliaikaista ja kummankin keskenään kommunikoivista järjestelmistä pitää olla toiminnassa. [6, s. 38-39.] Synkronisessa kutsussa järjestelmä jää odottamaan etäpään suoritusta, jolloin puhutaan järjestelmien välisestä tiukasta sidonnasta (tight coupling). Esimerkiksi tilanteissa, joissa vaaditaan tiettyä suoritusjärjestystä, voi synkronisuus olla kannattavaa.

Viestipohjaiset väliohjelmistot (MOM, Message Oriented Middleware) toteuttavat asynkronisuuden konseptia sanomajonojen avulla. Se mahdollistaa kommunikoinnin hajautettujen ja keskenään erilaisten alustojen välillä. Jono-pohjaisessa viestinvälityksessä sanoma lähetään tiettyyn välivarastoon, jonoon, josta etäpää voi lukea sanoman omaan tahtiinsa. Tällöin järjestelmä voi jatkaa omaa toimintaansa sanoman välityksen jälkeen,

vaikka etäpää ei olisi toiminnassa. Se siis mahdollistaa järjestelmien välisen löyhän kytkennän (loose coupling) [1, s. 95], eli järjestelmät pyritään pitämään mahdollisimman riippumattomina toisistaan [6, s. 39]. Tällöin järjestelmät toimivat itsenäisempinä komponentteina, eikä niiden tarvitse tuntea kohdejärjestelmää. [7.]

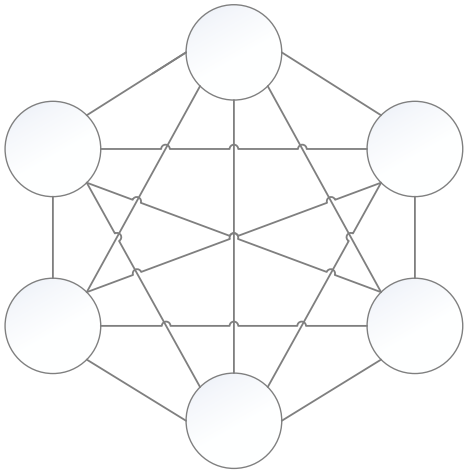
Löyhän kytkennän hyötyjen lisäksi väliohjelmisto tuo luotettavuutta viestinvälitykseen varastoimalla sanoman ja huolehtimalla, että sanoma välitetään vain kertaalleen. JMS (Java Messaging Service) on esimerkki tunnetusta avoimen standardin viestinvälitysprotokollasta, joka mahdollistaa jonojen käyttämisen. [7.] Esimerkkeinä viestipohjaisista väliohjelmistoista ovat mm. IBM Websphere MQ, Microsoft MSMQ ja Apache ActiveMQ. Asynkronisissa ratkaisuissa, joissa vaaditaan etäpäältä kuittausta viestin lukemisesta, puhutaan pseudo-synkronisesta tiedonsiirrosta [8].

2.2 Integraatioiden ratkaisumallit

Vuosien saatossa on kehittynyt useita erilaisia tapoja toteuttaa integraatioratkaisuja. Vanhemmat integraatioarkkitehtuurit eivät välttämättä kykene vastaamaan nykypäivän organisaatioissa tapahtuviin nopeisiin muutoksiin ja valtaviin järjestelmäkokonaisuuksiin. Seuraavaksi esitellään ratkaisumalleja integraatioiden toteuttamiseksi.

2.2.1 Point-to-Point

Järjestelmien liittäminen yhteen tapahtuu yksinkertaisimmillaan liittämällä ne suoraan toisiinsa, jolloin puhutaan Point-to-Point-integraatiosta (kuva 1). Point-to-Point, tai lyhyemmin P2P, on usein helpoin ja nopein tapa mahdollistaa järjestelmien välinen kommunikointi, ja sen käyttö on perusteltua tilanteissa, jossa integroitavia järjestelmiä on vain vähän.



Kuva 1. Kuuden järjestelmän Point-to-Point-integraatio

Järjestelmien lukumäärän kasvaessa yhteyksien määrä kasvaa eksponentiaalisesti. P2P-yhteyksistä puhuttaessa törmätäänkin usein Metcalfen lakiin, jonka mukaan verkon arvo on verkossa toisiinsa yhteyksissä olevien komponenttien neliö. Koska järjestelmän ei kuitenkaan tarvitse kommunikoida itsensä kanssa, ja yhteyksien oletetaan olevan kahdensuuntaisia, voidaan yhteyksien lukumäärä laskea alla olevalla yhtälöllä (kaava 1). [1, s. 22, 66-67; 3, s. 12.]

$$i = \frac{n(n-1)}{2} \quad (1)$$

i on yhteyksien lukumäärä
n on järjestelmien lukumäärä

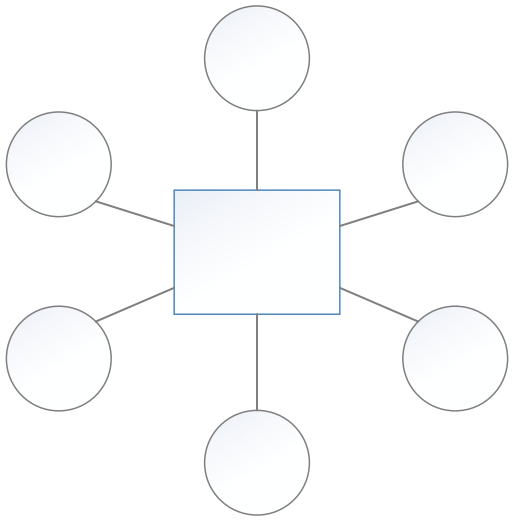
Esimerkiksi kuuden järjestelmän integrointi keskenään vaatisi maksimissaan 15 yhteyttä (kuva 1) ja sadan järjestelmän integrointi lähes 5000 yhteyttä. Toisin sanottuna uuden järjestelmän lisääminen vaatii olemassa olevien järjestelmien määrän verran uusia yhteyksiä. Vaikka todellisuudessa kaikkien järjestelmien ei välttämättä tarvitse olla yhteyksissä jokaiseen järjestelmään, kuten yllä oleva yhtälö olettaa, kuvaa se hyvin verkoston hallitsematonta kasvua.

Point-to-Point-integraatioiden vaarana onkin niin kutsuttu spagetti-integraatio, jossa integraatioarkkitehtuuri muodostuu liian vaikeasti ylläpidettäväksi ja valvottavaksi. Aluksi

kustannustehokas ratkaisu saattaa muuttua kalliiksi ratkaisuksi, sillä muutokset yhdessä järjestelmässä tai uuden järjestelmän lisääminen saattavat vaikuttaa myös muihin järjestelmiin. Kun järjestelmäkokonaisuus jatkaa kasvuaan, voidaan päätyä tilanteeseen, jossa muutostenhallinta alkaa olla mahdottomuus. Manouvrier [3, s. 12] kutsuu kirjassaan tätä tietojärjestelmän fossilisoitumiseksi (engl. fossilization).

2.2.2 Keskitetty integraatio

Integraatioiden suunnittelussa tulisi huomioida verkoston kasvu. Ylempänä huomattiin, että yhden komponentin lisääminen kokonaisratkaisuun kasvatti sen kokoa eksponentiaalisesti. Keskitetty kontrollipiste, johon järjestelmät ovat kytkeytyneet, mahdollistaa paremman skaalautuvuuden ja joustavuuden kuin P2P-integraatioarkkitehtuuri. Keskitetty integraatio, josta käytetään nimitystä hub-and-spoke, kasvattaa lineaarisesti vain yhden yhteyden jokaista uutta liitäntää kohden. (kuva 2) [1, s. 22,67.]



Kuva 2. Kuuden järjestelmän Hub and Spoke -integraatio

Hub and Spoke -integraatioarkkitehtuuri siis yksinkertaistaa integraatioita, kun kaikki komponentit ovat yhteydessä vain yhteen paikkaan. Tämä tarjoaa myös toisen hyödyn; keskittäminen mahdollistaa valvonnan ja kontrollin yhdestä paikasta. Sen heikkoutena on kuitenkin niin kutsuttu *single-point-of-failure*, jossa keskittimen (hub) ylikuormittuessa tai vikaantuessa kaikkien järjestelmien keskinäinen kommunikointi saattaa vaarantua. [1, s. 68.]

2.2.3 Palvelukeskeinen arkkitehtuuri ja palveluväylä

Palvelukeskeinen arkkitehtuuri (SOA, Service Oriented Architecture) on kehittynyt tarpeesta vastata aikaisempien integraatiomallien heikkouksiin. SOA:ssa ohjelmistot tarjoavat yhteiskäyttöisiä palveluita niitä kuluttaville sovelluksille. [2, s. 79.]

Palveluväylä (ESB, Enterprise Service Bus) on tekninen ratkaisu palvelukeskeisen arkkitehtuurin toteuttamiseen. [1, s.100.] ESB helpottaa eri teknologioilla toteutettujen palveluiden käyttöä tarjoamalla luotettavan, skaalautuvan ja tietoturvallisen viestintäinfrastruktuurin palveluiden välillä. ESB:n toiminnollisuuksiin kuuluu usein sanomien reititys kohteeseensa esimerkiksi viestin sisällön tai muiden tietojen avulla, sekä sanomamuunnosten tekeminen ennen viestin välitystä eteenpäin. Sen tarkoitus on siis lisätä joustavuutta palveluiden välillä sekä yksinkertaistaa integraatiota ja palveluiden uudelleenkäytettävyyttä. [2, s. 68-69.]

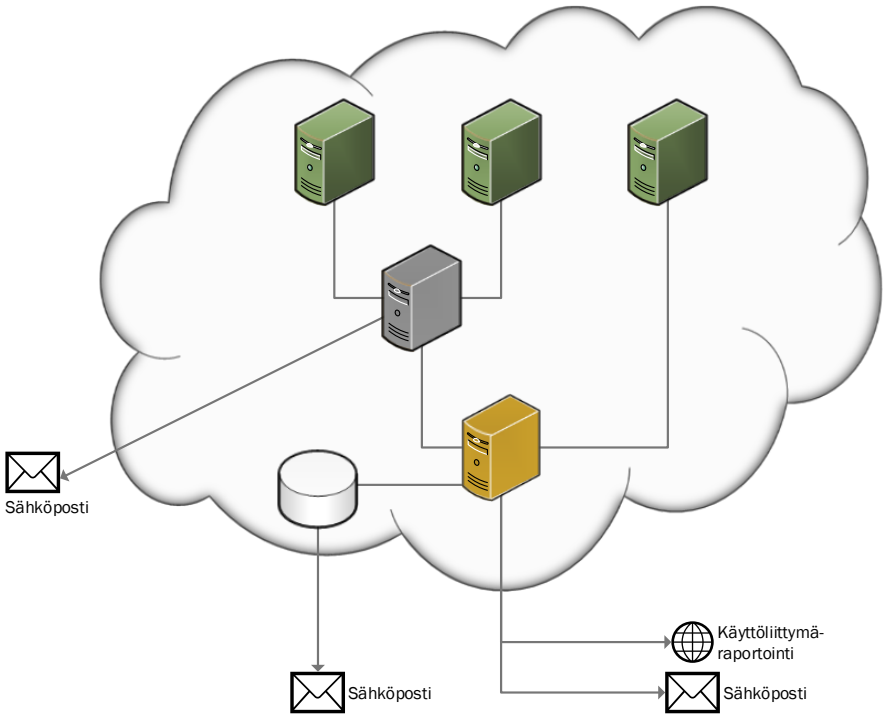
2.2.4 Integraatioiden tulevaisuus

Integraatioiden tulevaisuudesta puhuttaessa törmätään väistämättä keskusteluun mikropalveluarkkitehtuurista, jonka monet näkevät ESB:n korvaajana [9; 10]. Mikropalveluarkkitehtuurissa ohjelmistot jaetaan itsenäisiin komponentteihin, jotka toteuttavat yhtä tiettyä toimintoa ja kommunikoivat keskenään kevyillä, usein http-pohjaisilla, mekanismeilla [11]. Kaikki eivät kuitenkaan allekirjoita ESB:n hautaamista, vaan kummallekin teknologialle löytyvät omat käyttötarkoituksensa [12; 13].

3. Valvonnan nyky- ja tavoitetila

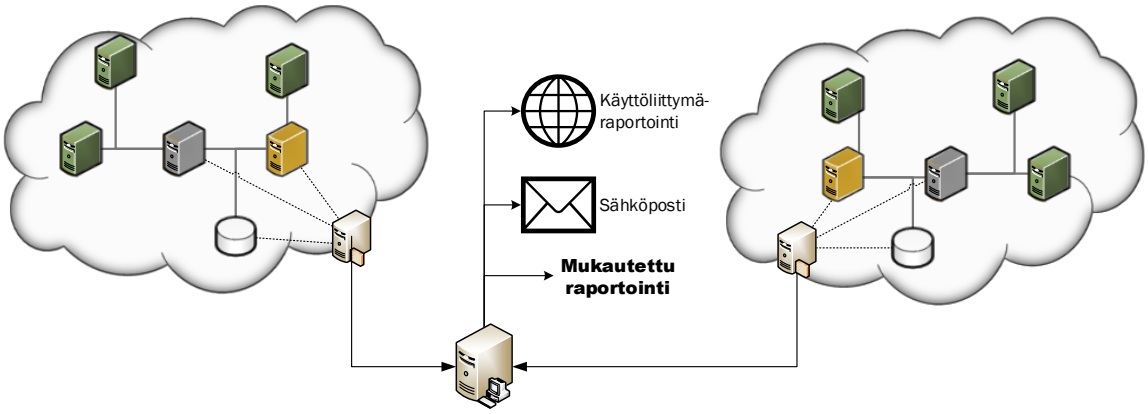
Kuten edellisessä luvussa huomattiin, järjestelmäintegraatioita voidaan toteuttaa monin eri tavoin riippuen organisaation tarpeista. Vaikka monet integraatio-ohjelmistot mahdollistavat tehokkaan ja ympäristökohtaisesti keskitetyn valvonnan, tarvitaan moniasiakasympäristöissä laajempaa valvontakeskittymää. Tällä mahdollistetaan nopea reagointi ongelmatapauksissa pienemmillä resursseilla, kun tieto löytyy yhdestä paikasta. On myös tarpeen valvoa integraatioiden kannalta kriittisiä verkkoinfrastruktuurin komponentteja, joiden valvontaa integraatio-ohjelmistot eivät yleensä laajamittaisesti tarjoa.

Työssä kehitettävä valvontaratkaisu otetaan alkuun käyttöön osalle valvottavista integraatioista. Valvontaratkaisun kehityksen ei tulisi haitata jo olemassa olevaa valvontaa. Kuvassa 3 on nykytilanne esimerkkiympäristöstä. Vaikka valvottavat ympäristöt ovat keskenään erilaisia, on syytä huomata kokonaisratkaisun eri komponenttien huolehtivan valvonnasta itsenäisesti.



Kuva 3. Esimerkkiympäristö, jossa komponentit hoitavat valvontansa itsenäisesti

Esimerkkiympäristö (kuva 3) sisältää erilaisia järjestelmiä ja niiden sanomanvälitykseen erikoistuneita komponentteja, kuten jonopalvelimen ja integraatioalustan. Tällä hetkellä osa valvonnasta tapahtuu suoraan valvottavissa ympäristöissä: integraatio-ohjelmistojen ja muiden komponenttien omilla valvontatyökaluilla ja mukautetuilla skripteillä. Tavoitetilassa (kuva 4) pääasiallinen valvonta siirtyisi keskitetylle valvontaratkaisulle, joka nopeuttaisi ongelmiin reagointia ja vapauttaisi resursseja.



Kuva 4. Ympäristö, jossa valvonta on keskitetty

Kuvassa 4 eri ympäristöjen komponentit välittävät välityspalvelimien kautta valvonta-dataa keskitetylle valvontapalvelimelle, joka prosessoi datan ja huolehtii raportoinnista ja hälytyksistä. Järjestelmät voivat myös olla suoraan yhteydessä valvontapalvelimeen.

Olemassa olevaa valvontaa tulee voida myös suorittaa karsitusti, jolloin ei päädytä valvonnassa ns. single-point-of-failure-tilaan, jossa valvontaratkaisun häiriön vuoksi menetettäisiin valvonta, vaan voidaan käyttää hetkellisesti vanhaa valvontaratkaisua. Myös valvontapalvelimien kahdennuksilla voidaan varautua vikatilanteisiin. Koska integraatiot suunnitellaan laajennettavuutta silmällä pitäen, tulee valvonnassakin kiinnittää erityistä huomiota skaalautuvuuteen. Valvontaratkaisun valinnassa ja kehityksessä tulisi huomioida myös tulevaisuuden tarpeet ja varmistaa, että ratkaisu toimii ja on hallittavissa myös suuremmissa ympäristöissä.

4. Valvontaratkaisu

”*Vain pieni osa järjestelmäintegraatioratkaisun suunnittelusta on toiminnallisuuteen liittyvää, loput suunnittelupanoksista käytetään varautumiseen mahdollisiin tai mahdottomiin vikatilanteisiin.”* [1, s. 107.]

Tässä luvussa pohditaan, mitä tavoitetilaan pääseminen vaatii, mitä ominaisuuksia valvontaratkaisulta halutaan ja kerrotaan esimerkkejä eri valvontaratkaisuista, joilla nämä kriteerit täyttyisivät sekä eritellään syitä valvontaratkaisun tekemiselle. Lopuksi esitellään kaksi esimerkkiä integraatiolle oleellisten valvontakohteiden valvonnasta ja määritellään

niiden onnistumiskriteerit, joiden avulla tutkitaan valvontaratkaisun onnistumista integraatiovalvonnan näkökulmasta.

4.1 Valvontaratkaisun kriteerit

Valvontaratkaisun valintaan tulee määritellä kriteerit, joiden pohjalta voidaan valita ohjelmisto, joka tarjoaa tarvittavat toiminnallisuudet. Valvontaratkaisun voidaan olettaa tekevän seuraavat asiat [14, s. 25]:

- Dataa voidaan kerätä eri lähteistä.
- Kerättyä dataa voidaan verrata raja-arvoihin ja hälytyksiä voidaan lähettää tarvittaessa esimerkiksi sähköpostilla.
- Dataa voidaan säilyttää myöhempää analysointia varten.
- Dataa voidaan visualisoida kerätyn datan pohjalta.

Erilaisia valvontaohjelmistoja, jotka tarjoavat edellä mainitut ominaisuudet, löytyy monia. Verkkomonitorointiin perustuvia ohjelmistoja ovat muun muassa Zabbix, Nagios, Pandora FMS ja Solarwinds. Myös lokistatistiikkatyökaluja voidaan käyttää hyvin monipuoliseen valvontaan. Zabbix valikoitui valvontaohjelmistoksi muun muassa aikaisemman osaamisen lisäksi. Vahva Zabbix-yhteisö helpotti tiedon löytymistä internetistä. Myös avoin lähdekoodi ja kaikkien ominaisuuksien käyttäminen maksutta vaikuttivat päätökseen.

4.2 Syyt valvonnalle

Asiakkaan ja toimittajan tulee voida tarkasti määritellä, onko palvelun taso halutun kaltaista. SLA (Service Level Agreement) eli palvelutasosopimus määrittelee, mitä asiakas vaatii tilaamaltaan palvelulta. SLA:n sopimuskohtien tulee olla laskettavia suureita, jolloin voidaan täsmällisesti laskea, onko toimittaja päässyt luvattuihin tavoitteisiin. Asiakkaan ja järjestelmäintegraatiotoimittajan välisessä SLA-sopimuksessa on yleisesti mainittuna mm. seuraavia asioita:

- Vastausaika (Mean time to response) määrittää, kuinka kauan toimittajalla on keskimäärin kulunut aikaa palvelupyyntöön vastaamiseen tai työn alle ottamiseen.

- Ratkaisuaika (Mean time to resolve) määrittää, kuinka kauan palvelupyynnön ratkaisuun on kulunut aikaa.

- Palveluaika kertoo ajat, jolloin ylläolevat kohdat ovat voimassa, eli milloin toimittajan tulee olla tavoitettavissa.

- Saatavuus kertoo, kuinka suuren osan ajasta tietty palvelu, esimerkiksi palvelin tai integraatioalusta, on saatavilla.

- Sanktiot määrittävät toimittajalle aiheutuvat sanktiot tilanteessa, jossa ei ole päästy yllä oleviin tavoitteisiin.

Valvontaratkaisun tulisi auttaa vastaus- ja ratkaisuaikojen parantamisessa tunnistamalla virheet ajoissa ja tarkasti. Valvonnan tuottamia eri metriikoita tulisi voida korreloida keskenään ja saada tilanteesta suurempi kokonaiskuva. Valvontaratkaisun tulisi luonnollisesti auttaa myös virheiden ennaltaehkäisyssä esimerkiksi varoittamalla ajoissa raja-arvojen ylityksistä. Asiakaslähtöisellä ajatusmallilla toimittajan tulee voida vastata myös määrittelemättömiin asiakkaan tarpeisiin. Asiakastyytyväisyyden ja asiakkaiden tarpeiden tulisi olla keskiössä, eikä lähtökohtaisesti asiakkaan tulisi huomata virhettä ennen toimittajaa.

Valvontaratkaisun kehityksessä pyritään edistämään palvelunhallintaa. Valvontaratkaisun tulee voida virheenselvityksen lisäksi tunnistaa myös virheiden vakavuudet. Kun tikettijärjestelmään integroitu valvontaratkaisu luo tiketin, tulee sen myös helpottaa tiimiä ongelmien priorisoinnissa, näin ollen kriittisten ongelmien selvitys nopeutuu, kun valvontajärjestelmän määrittelyjen perusteella on asetettu tiketille jo kriittisyysaste. Toimittajalle valvontaratkaisu tuo myös muita hyötyjä. Kun ongelmiin voidaan reagoida nopeammin ja tehokkaammin, vapautuu tiimiltä resursseja muiden ongelmien selvityksiin. Toimittaja pystyy myös lupaamaan jatkossa kustannustehokkaampia ja laadukkaampia palveluja sekä sopimuksia.

Valvontaratkaisuiden tulisi myös olla yleiskäyttöisiä, jolloin niitä voidaan helposti käyttää muissa ympäristöissä konfiguroimalla valvonnat järjestelmän tarpeiden ja sopimuksien mukaan.

4.3 Valvontakriteereiden taulukointi ja onnistumisen määrittely

Valvontametodit ja onnistumiskriteerit vaihtelevat ympäristökohtaisesti. Alla oleviin taulukoihin on koottu erään ympäristön kaksi integraatioiden kannalta oleellista valvontakohdetta, jotka tulee saada valvontaan. Miksi-kohdassa on lyhyt selitys kyseisen valvontakohteen tarpeelle. Raportoinnin tulisi lähtökohtaisesti tapahtua sähköpostitse palvelun ylläpitäjän tikettijärjestelmään, jonka vastuulla on luoda tapahtumasta, joko tiketti tai ilmoitus, sekä määrittää tiketille SLA:n mukaiset määreet, kuten vaste- ja palveluajat. Ilmoituksissa ja ei-kriittisissä hälytyksissä riittää, että valvontajärjestelmä ilmoittaa tilanteesta esim. käyttöliittymän hallintapaneelissa.

Kriteerit-kohta määrittelee valvonnan onnistumiskriteerit, jotka kyseisen valvonnan tulee minimissään suorittaa. Kriteerit ovat kehitettävän valvonnan kriteereitä, ei integraatiokehityksen kriteereitä. Esimerkiksi integraatiokehityksen onnistumiskriteerinä voisi olla, ettei DeadLetter-jonoon saa mennä sanomia. Valvontaratkaisun onnistumiskriteerinä voisi tällöin olla hälytys kyseisestä tapahtumasta. Onnistumiskriteereiden täyttymistä tarkastellaan luvussa 7.

Taulukko 1. Jonovalvonnan onnistumiskriteerit

Jonojen syvyyksien valvonta	
Miksi	Jonoihin jäävät viestit ovat todennäköinen merkki siitä, ettei kohdejärjestelmä ole lukenut niitä eivätkä viestit ole välittyneet määränpäähän. Dead letter -jonoon saapuvat viestit ovat merkki erilaisista virhetilanteista.
Raportointi	Kriittisistä virheistä tulee välittää tieto palvelun ylläpitäjälle. Ilmoitustasoisista herätteistä tulee olla saatavilla tieto esimerkiksi hallintapaneelissa.

Kriteerit	1. Asiakkaan A jonopalvelimen X DeadLetter-jonoon tulee sanoma. Valvontajärjestelmä lähettää virheen sähköpostitse tiimille 15 minuutin kuluessa.
	2. Asiakkaan A jonopalvelimelle X jonon Q syvyys kasvaa arvoon 250. Valvontajärjestelmä ilmoittaa tilanteesta käyttöliittymällä.
	3. Asiakkaan A jonopalvelimelle X jonon Q syvyys kasvaa arvoon 400. Valvontajärjestelmä lähettää ilmoituksesta sähköpostia.

Taulukko 2. Hiljaisuudenvalvonnan onnistumiskriteerit

Sanomien hiljaisuuden valvonta	
Miksi	Tietyissä tietovirroissa voidaan olettaa sanomien kulkeminen säännöllisesti. Hiljaisuuden valvonnan tarkoitus on ilmoittaa, mikäli jossakin tietovirrassa ei kulje sanomia tiettyinä aikoina.
Raportointi	Kriittisistä virheistä tulee välittää tieto palvelun ylläpitäjälle sähköpostitse. Ilmoitustasoisista herätteistä tulee olla saatavilla tieto, esim. hallintapaneelissa.
Kriteerit	1. Kriittisessä tietovirrassa ei ole kulkenut sanomia määrättynä ajanhetkenä. Valvontajärjestelmä lähettää sähköpostia.

Taulukoissa esitetyt valvonnat ovat esimerkkejä integraatioille tyypillisistä valvonnoista. Verkkoinfrastruktuurivalvontaakin kehitetään integraatiovalvonnan ohessa ja niiden onnistumista tarkastellaan työssä yleisellä tasolla.

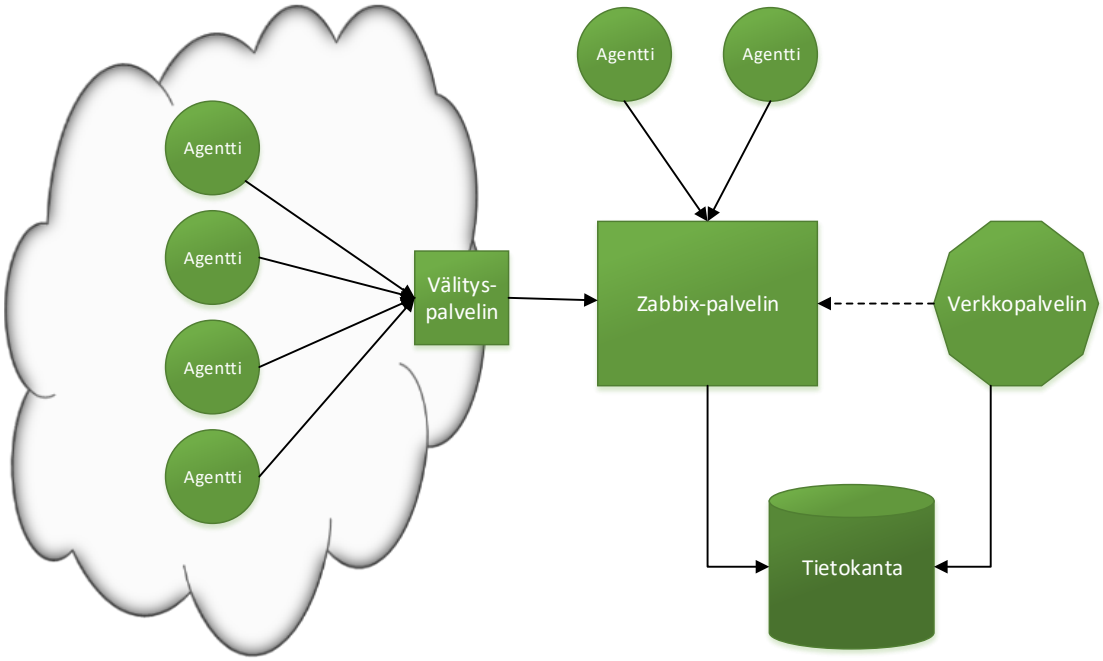
5. Zabbix

Zabbix on ilmainen ja GPLv2-lisensoitu avoimen lähdekoodin valvontatyökalu yritysten IT-infrastruktuurin valvontaan. Zabbix ei sisällä erikseen yritysversioita, eikä sillä ole rajoituksia valvottavien kohteiden tai datan käytön suhteen. [15.] Se on valvontasovellus, jonka käyttö ei rajoitu vain verkkoinfrastruktuurin valvontaan, vaan valvonnan voi konfiguroida koskemaan lähes mitä tahansa valvontaa. Se tarjoaa valvontasovellukselle

oleelliset toiminnot, kuten datan keräämisen, datan säilytyksen, hälytykset sekä erilaisia raportointitapoja [14, s.25]. Zabbixin monitorointi, konfigurointi ja kontrollointi tapahtuu pääasiallisesti graafisen web-käyttöliittymän avulla. [16, s. 11.]

5.1 Pääkomponentit

Zabbix vaatii minimissään Zabbix-palvelimen, tietokannan ja web-palvelimen. Nämä voivat olla pienempien järjestelmien tapauksessa yhdellä palvelimella, tai suuremmissa ympäristöissä eri palvelimilla. Näiden kolmen komponentin lisäksi on kaksi datan keräykseen ja välitykseen erikoistunutta komponenttia, agentit ja välityspalvelimet (proxy), joiden kautta haluttu data saadaan. [14, s. 27-28.] Kuvassa 5 on yksinkertainen kuvaus näiden viiden eri komponentin yhteyksistä.



Kuva 5. Zabbixin pääkomponenttien yhteydet

Kuvassa 5 agentit keräävät dataa kohdejärjestelmistä. Kerätty data lähetetään Zabbix-palvelimelle joko suoraan tai välityspalvelimen kautta. Välityspalvelimella voidaan mahdollistaa pääsy aliverkkoon esimerkiksi palomuurin lävitse tai sitä voidaan käyttää kuormanjakana. Zabbix-palvelin tallentaa kerätyn datan tietokantaan, jota web-palvelin lukee

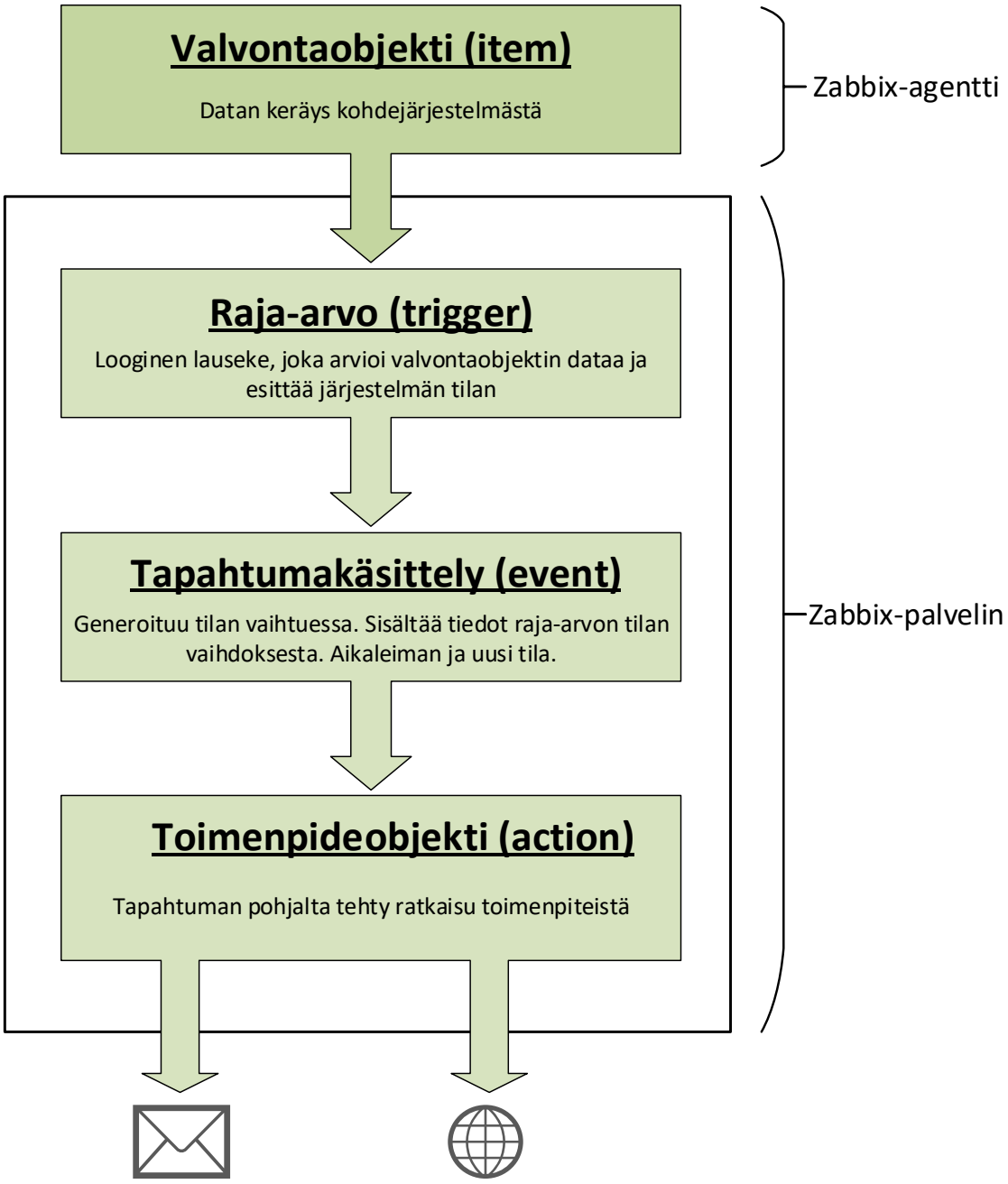
ja näyttää tiedon käyttäjälle. Web-palvelin on yhteydessä Zabbix-palvelimeen, jos halutaan saada tietoa palvelimen tilasta.

Zabbix-palvelin voidaan asentaa useimmille Unix-tyyppisille palvelimille, kuten Linuxille, AIX:lle ja Mac OS X:lle. Valvottavaan kohteeseen asennettava Zabbix-agentti toimii myös uudemmissa Windows-käyttöjärjestelmissä. Ulkopuolisten verkkopalveluiden valvontaa voidaan suorittaa ilman agenttia. [17.] Tässä luvussa käydään läpi tarkemmin Zabbixin komponentit ja käyttötarkoitukset.

5.2 Tiedonkeruu ja objektit

Monissa valvontasovelluksissa tietoa keräävä agentti käsittelee tiedon ja lähettää tilannetiedon palvelimelle. Zabbix poikkeaa tästä käsittelytavasta siten, että agentti lähettää raakadataa suoraan Zabbix-palvelimelle, joka käsittelee tiedon. Eli valvottavalle kohteelle asennettu agentti ei tee itse päättelyä keräämästään datasta. Se ei siis päätä esimerkiksi, onko suorittimen käyttö yli raja-arvon tai onko muisti vähissä ja lähetä palvelimelle OK tai ERROR-ilmoitusta, vaan se välittää palvelimelle vain mittausarvoja. Kun data erotetaan käsittelyprosessoinnista, saadaan kaksi merkittävää etua. Ensinnäkin, voidaan kerätä dataa ilman, että sille määritellään käsittelyä. Voi olla, että dataa halutaan vain visualisoida tai varastoida. Toinen etu on mahdollisuus korreloida dataa keskenään. Voidaan hyödyntää keskitettyä tietokantaa ja käyttää päätöksenteon apuna historiallista dataa tai vaikka toisen järjestelmän dataa, joka saattaa vaikuttaa haluttuihin toimenpiteisiin. [16, s. 120-121.]

Yksinkertaistetusti datan keräyksen ja raportoinnin väliset loogiset toimenpiteet voisi kuvata alla olevan kuvan (kuva 6) mukaan.



Kuva 6. Datan keräyksen ja raportoinnin väliset toimenpiteet

Kuvassa 6 kerätty data lähetetään Zabbix-palvelimelle käsiteltäväksi. Agentti kerää kohdejärjestelmästä mitattavaa tietoa, jonka pohjalta Zabbix-palvelin päättelee, onko mitattava tieto yli määritettyjen raja-arvojen. Mikäli raja-arvot ylittivät tai alittuvat raja-arvo-objektin tila muuttuu tapahtumakäsittelyssä ja Zabbix-palvelin tekee määritellyt toimenpiteet, esimerkiksi lähettää sähköpostia virhetilanteesta tai näyttää virheen käyttöliittymällä.

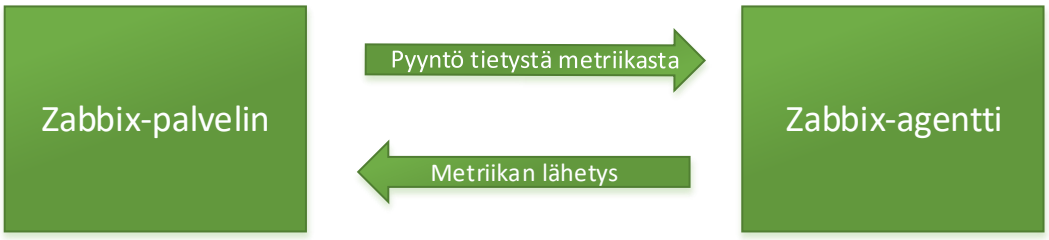
5.2.1 Valvontaobjekti

Valvontaobjekti (item) kuvaa yksittäistä arvoa. Suorittimen käyttöaste tietyllä ajanhetkellä tai shell-skriptin tulostus ovat esimerkkejä valvontaobjektin arvoista. Sille määritellään tunnistetieto, tietotyyppi ja palvelin, johon se on liitetty. Tunnistetieto koostuu nimestä, jonka tarkoitus on olla mahdollisimman kuvaava nimi, sekä avaimesta, joka on halutun toimenpiteen tarkka syntaksi. Tunnistetiedon ja palvelimen nimen avulla valvontaobjekti voidaan erottaa muista valvontaobjekteista. Zabbix käyttää tietotyyppiarvoa päättelyssä, miten tieto varastoidaan, visualisoidaan, ja mitä tiedolla voidaan ylipäätään tehdä. [16, s. 122.]

Valvontaobjektille myös määritellään, kuinka kauan aikaa historiaa halutaan säilyttää. On tarpeen miettiä sopivat tallennusajat, jottei resursseja hukata tarpeettoman pitkien historiajaksojen säilyttämiseksi. Historiallisen datan säilytystä voidaan hyödyntää tehokkaasti esimerkiksi tiettyjen valvonta-arvojen kehityssuunnan ennustamisessa. [16, s. 53, 122.]

Valvontaobjektin tyypillä määritellään, kuinka tieto saadaan valvottavalta kohteelta Zabbix-palvelimelle. Agenttipohjaisen valvonnan lisäksi voidaan valvontaa suorittaa ilman agenttia, esimerkiksi ulkoisten skriptien avulla, joista lisää myöhemmin. [16, s. 122.]

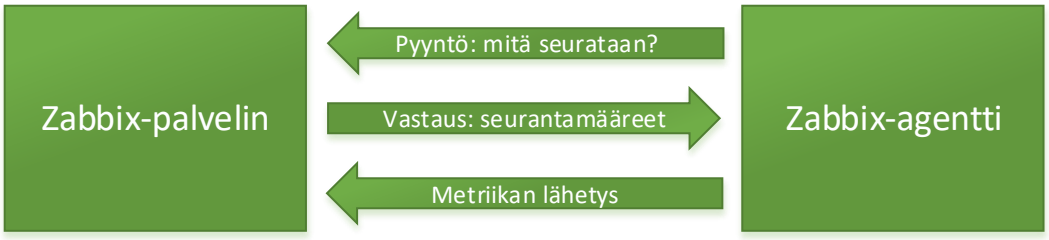
Zabbixissa valvontaobjekti voi olla aktiviinen tai passiivinen. Oletusarvoisesti se on passiivinen (kuva 7), jolloin palvelin kysyy agentilta tietyin väliajoin mittaustuloksia, jotka agentti lähettää välittömästi. Agentti siis jatkuvasti kuuntelee palvelimelta tulevia pyyntöjä. [16, s. 123.]



Kuva 7. Passiivinen valvontaobjekti [18, mukaillen kuvaa passive mode]

Passiivisen valvontaobjektin hyötyjä ovat mm. asennuksen ja virheenselvityksen helppous sekä joustavasti asetettavat ajastukset. [18.]

Aktiivisen valvontaobjektin (kuva 8) tapauksessa agentti kysyy palvelimelta, millaista dataa sen tulisi kerätä ja kuinka usein. Sen jälkeen se ajastaa mittauksensa ja yhdistää palvelimelle datan lähettämistä varten [16, s. 123].



Kuva 8. Aktiivinen valvontaobjekti [19, mukaillen kuvaa active mode]

Aktiivinen valvontaobjekti on kätevä ratkaisu esimerkiksi silloin, kun ei haluta sallia sisään tulevaa liikennettä valvottavalle koneelle, vaan halutaan sallia vain ulospäin kulkeva liikenne. Se vähentää myös Zabbix-palvelimen kuormitusta ja mahdollistaa NAT:n käytön. [18.] Sekä passiivista että aktiivista valvontaobjektia voidaan käyttää samalla agentilla. [16, s. 124.]

5.2.2 Raja-arvo

Raja-arvo-objektit (trigger) kuvaavat määritetyn valvonnan tilaa. Ne arvioivat valvontaobjektien keräämän datan perusteella, onko jokin valvontamääritys sallittujen arvojen puitteissa. Esimerkiksi raja-arvoksi voidaan määrittää, että kun levytilan käyttöaste on tietyn rajan yli, luo se tapahtumakäsittelijän (event), joka sisältää tiedot raja-arvon ylityksestä. [16, s. 222-223.]

5.2.3 Toimenpide

Hälytykset ovat tärkeitä kaikissa monitorointijärjestelmissä. Zabbixissa hälytykset määritellään toimenpideobjektilla (action). Toimenpideobjektit ovat globaalisti määriteltyjä, eikä tiettyyn valvottavaan kohteeseen, jolla mahdollistetaan, ettei tarvitse käsitellä vain yhtä valvontaobjektia ja määrittää sille raja-arvoa. Se voi käyttää hyväkseen koko tietokantaansa ja tehdä monimutkaisia päätöksiä hälytyksille useiden valvontaobjekteiden, tapahtumien tai valvottavien palvelimien pohjalta. [16, s. 211, 222.]

Toimenpideobjektin toiminnallisuus koostuu kolmesta osasta: määrityksistä, ehdoista ja operaatioista. Määrityksiin kuuluvat esimerkiksi toimenpiteen nimi ja oletusviesti, joka halutaan lähettää. Ehdoissa kuvataan, minkä tapahtuman ja valvontakohteen yhteydessä tehdään haluttu operaatio. Erilaisia ehtoja voidaan yhdistää loogisilla operaattoreilla yhteen halutuilla perusteilla.

Toimenpiteen operaatioihin voidaan määritellä toimenpiteen suoritus ja mitä tehdään, jos tilanne paranee normaaliksi. Operaatioita on esimerkiksi sähköpostin tai tekstiviestin lähettäminen, mutta myös omia mediatyyppejä voidaan luoda ja integroida hälytykset vaikka pikaviestisovelluksiin tai tikettijärjestelmiin. Eri henkilöille voidaan myös määritellä erilaiset viestit. Operaatioiksi voidaan myös määrittää etäkomentojen suorittaminen esimerkiksi agentin tai SSH-yhteyden kautta. [16, s. 223-227.]

5.2.4 Zabbix-välityspalvelin

Zabbix-välityspalvelin (proxy) on Zabbix-ohjelmistoperheeseen kuuluva komponentti, joka voidaan laittaa Zabbix-palvelimen ja Zabbix-agentin väliin keräämään ja säilyttämään dataa valvottavista kohteista omassa tietokannassaan. Agentin tavoin sillä ei ole omaa käyttöliittymää, vaan se on yksinkertainen ja kevyt työkalu välikappaleeksi esimerkiksi palomuurin takana oleville järjestelmille. Välityspalvelimen tarkoitus on kerätä data valvottavilta kohteilta yhteen eikä prosessoida tätä monimutkaisilla hauilla tai laskutoimituksilla. Tämän kuuluisi olla helppo ja edullinen ratkaisu, sillä se ei vaadi paljoa prosessointitehoa tai levytilaa, sillä data prosessoidaan Zabbix-palvelimella. Välityspalvelimelle riittäneekin yleensä pieni tietokanta ja tietokannoiksi soveltuu samat tietokannat kuin Zabbix-palvelimelle. [16, s. 56-57.] Proxyjen käyttö valvontaratkaisussa ei kuitenkaan ole pakollista, eikä aina edes kannattavaa varsinkaan pienempien ympäristöjen kohdalla.

5.2.5 Tietokantojen monitorointi

Zabbixilla on mahdollisuus tehdä tietokantoihin SQL-kyselyitä. Tietokantakyselyn vastaus tallennetaan valvontaobjektin arvona ja niihin voidaan liittää raja-arvot. Kyselyitä voi suorittaa mihin tahansa ODBC:tä (Open Database Connectivity) tukeviin tietokantoihin. Valvontaobjektin tyypiksi valitaan tietokantavalvonta, asetetaan tietokannan nimi, käyttäjätunnus, salasana ja haluttu kysely. Kysely tulee aina alkaa select-lausekkeella ja sen tulee palauttaa vain yksi arvo. [16, s. 126-135.]

5.2.6 JMX-monitorointi

JMX (Java management Extensions) tarjoaa tavan sovellusten sekä laitteiden hallintaan ja valvontaan. Versiosta 2.0 lähtien Zabbix mahdollistaa JMX:n hyödyntämisen Zabbix Java-yhdyskäytävän (Zabbix Java Gateway) avulla. Kyselyt suoritetaan Oraclen JMX-hallinta API:n kautta. Valvottavalla sovellus tulee olla käynnistettynä JXM remote console -toiminto käytössä. Koska JMX:n avulla sovelluksia voidaan hallita, on tarpeen suunnitella käyttö huolella, etteivät ulkopuoliset pääse käsiksi sovellukseen. [16, s. 137-138.]

5.2.7 Mallit

Suuremmissa Zabbix-ympäristöissä saattaa olla paljon eri objekteja, kuten valvontaobjekteja, kaavioita, toimenpideobjekteja ja niin edespäin. Näiden konfiguraatio olisi lähes mahdotonta toteuttaa ilman jonkinlaista yhtenäistämistä. Malleilla (template) voidaan määrittää eri objekteista kokoelmia koskemaan useita samankaltaisia valvottavia palvelimia.

Löytö-toiminnallisuus (discovery) luo säännöt Zabbixille, jonka avulla se tunnistaa uudet valvottavat palvelimet ilman manuaalisia luomisia. Toiminnollisuudelle määritellään ip-alue, jonka sisältä palvelin etsii agentteja määrätyin väliajoin. Löytäessään uuden agentin löytötapahtuma (discovery event) generoituu, ja toimenpideobjektilla määritetyt toimenpiteet, kuten palvelimen liittäminen tiettyihin malleihin, tapahtuvat. [16, s. 233, 245-247.]

5.2.8 Ulkoiset kyselyt ja käyttäjäparametrit

Zabbix tarjoaa kattavasti erilaisia natiiveja valvontaobjekteja, joita voidaan helposti ottaa käyttöön ilman, että tarvitsee ymmärtää syvempää toimintaa valvonnan takana. Kaikkiin mahdollisiin valvontamäärityksiin ei kuitenkaan valmiita objekteja ole. Ulkoinen kysely (external check) on Zabbix-palvelimelle kirjoitettu skripti, jota Zabbix ajaa kohteessa. Ulkoinen kysely ei vaadi agenttia valvottavaan kohteeseen. Koska ulkoiset kyselyt ovat Zabbix-palvelimella prosessoitavia, voivat ne kuormittaa palvelinta. Käyttäjäparametri (user parameter) on agentin konfiguraatioon kirjoitettu komento, jota agentti suorittaa valvottavalla palvelimella määrätyin väliajoin. Käyttäjäparametrien käyttö jakaa kuormaa kohdepalvelimille Zabbix-palvelimen sijasta. [16, s. 253.]

6. Valvonnan kehittäminen Zabbixilla

Tässä luvussa kerrotaan projektin alkutilanteesta sekä Zabbix-valvonnan suunnittelusta ja toteutuksesta. Koska työ on rajattu käsittelemään valvonnan tehostamista, ei tässä käydä läpi Zabbixin asennusta tai agenttien konfigurointia tarkemmin. Luvussa kerrotaan muun muassa valvonnan suunnittelusta, kehityksestä sekä kahden esimerkkivalvonnan toteutustavasta. Esimerkkivalvontana toteutetun jonovalvonnan tarkempi tekninen toteutusesimerkki löytyy liitteestä 1.

Zabbix-palvelin, tietokanta ja käyttöliittymä sijaitsivat toimittajan verkossa yhdellä Linux-palvelimella. Palvelimelle oli asennettuna Zabbixista versio 3.2, jonka tietokantana oli käytössä MySQL. Valvottavat järjestelmät sijaitsivat asiakkaiden omissa verkoissa tai toimittajan keskitetyssä iPaaS (Integration Platform as a Service) -ratkaisussa. Valvottaville järjestelmille oli asennettu Zabbix-agentit, joiden kautta valvontaa suoritetaan.

Teknistä velkaa syntyy kiireessä. Teknistä velkaa syntyy yleensä, kun halutaan nopeita ratkaisuja ylläpidettävyydestä piittaamatta. Tällöin uusien ominaisuuksien tekeminen muodostuu optimitilannetta haasteellisemmaksi ja velkaa maksetaan kasvaneilla ylläpito- ja kehityskustannuksilla. Suunnittelussa tulisi siis huomioida valvontaratkaisun jatkuva kasvaminen ja ominaisuuksien uudelleenkäytettävyys.

Kehityksessä tuli ottaa huomioon tietoturva ja erilaiset verkkoratkaisut. Valvottavien kohteiden osalta tuli suunnitella, mitä dataa on tarpeellista kerätä, kuinka usein valvontadataa tuli kerätä, kuinka kauan historiaa tulee säilyttää, mitkä ovat raja-arvot valvottavalle datalle, onko datalla yhteyksiä muuhun dataan ja kuinka hälytykset tulisi suorittaa.

Hälytykset päätettiin suorittaa tikettijärjestelmään sähköpostiviestien avulla. Zabbixille luotiin käyttäjä, jonka sähköpostiosoite vastaa tikettijärjestelmän osoitetta. Näin käyttäjälle kohdistetut hälytykset saatiin tiketöityä automaattisesti. Vähemmän kriittisissä virheissä riitti ilmoitus Zabbixin käyttöliittymällä. Käyttöliittymälle tuli automaattisesti tieto kaikista raja-arvojen ylityksistä, joten se ei vaatinut omaa toimenpidettä. Käyttöliittymähälytykset voitiin Zabbix 3.2 versiosta eteenpäin merkitä käsitellyiksi suoraan käyttöliittymältä, mikä helpotti virheilmoitusten ja -hälytysten käsittelyä.

Zabbixissa voidaan käyttää sertifikaattipohjaista ja esijaetun avaimen salauksia. Esijaetun avaimen salauksessa generoidaan 256-bittinen salausavain, joka asetettiin Zabbix-palvelimelle ja agentille salaamaan niiden väliset viestit. Toinen tietoturvaa lisäävä tekijä on aktiivisten valvontaobjektien käyttö, jolloin Zabbix-palvelimen ei tarvitse olla yhteydessä valvottavaan kohteeseen. Tällöin asiakasympäristöstä ei tarvitse avata portteja sisäänpäin liikenteelle. Aktiivisen valvontaobjektin asettamiseksi agentin konfiguraatiossa asetettiin Zabbix-palvelimen osoite aktiivista tiedonsiirtoa varten. Zabbix-palvelimella valvontaobjektien tyypeiksi määriteltiin aktiivinen agentti.

Kun Zabbix palvelin ja agentti oli konfiguroitu, voitiin valvonta toteuttaa yksinkertaistetusti seuraavilla askeleilla:

- Luodaan valvontaobjekti Zabbix-palvelimelle asettamalla kerättävän datan informaation tyyppi, päivitysintervalli, historian säilytystiedot ja mahdolliset muut valvontaobjektin asetukset.

- Asetetaan agentille sen mahdollisesti suorittamat komennot, kuten käyttäjäparametrit, esimerkiksi skripti, joka tekee tietokantakyselyn.

- Luodaan raja-arvo-objekti Zabbix-palvelimelle, jossa määritellään valvonnan raja-arvot.

- Luodaan toimenpiteet Zabbix-palvelimelle, jossa määritellään hälytystoiminta, esimerkiksi sähköpostin lähettäminen tikettijärjestelmään.

Valvonnan voi konfiguroida koskemaan tiettyä kohdetta tai mallien avulla useita samankaltaisia kohteita. Koska kehityksessä tuli suosia uudelleenkäytettävyyttä, niin lähtökohtaisesti kaikki valvonta tapahtui mallien avulla. Zabbixissa on valmiina useita malleja, jotka sisältävät valmiita valvontamäärityksiä erilaisien natiivien objektien avulla. Esimerkiksi eri käyttöjärjestelmille oli olemassa valmiit mallit, jotka sisälsivät käyttöjärjestelmäkohtaiset valvonnat levytilan kulutukselle, prosessorin käyttöasteelle, palvelimen pingauksen ja muita yleiseen valvontaan liittyviä määreitä.

Valvottaville Linux-palvelimelle voitiin esimerkiksi asettaa malli *Template OS Linux Active*, jolle oli luotu Linux-palvelimille ominaisia aktiivisia valvontaobjekteja, raja-arvoja ja toimenpiteitä. Kehitetyt valvontamääritykset pystyttiin laittamaan suoraan mallille ja

kaikki mallia käyttävät kohteet saivat uudet määritykset. Kohteella pystyi olemaan useita malleja, joten esimerkiksi tietokantapalvelimelle voitiin asettaa kohteen käyttämä käyttöjärjestelmä- ja tietokantamalli.

Jonovalvonta

ActiveMQ-jonopalvelimia varten luotiin malli, *Template App ActiveMQ Active*, johon määriteltiin ohjelmistokohtaiset valvontamääritykset. Tavoitteena oli saada malliin liitettyä esimerkiksi Dead letter -jonon ja muiden haluttujen jonojen syvyydet, virhelokikirjaukset, väliohjelmiston saatavuustiedot, muistinkäyttö ja muut ohjelmistoon liittyvät valvontamääritykset. Jonosyvyyksien määrittely noudatti keskenään samaa kaavaa, joten tässä esitellään vain Dead letter -jonovalvonnan määrittely, johon viestit päätyvät erilaisissa virhetilanteissa. Esimerkiksi jos pyritään lähettämään viestiä olemattomaan jonoon, jono on täynnä tai viestin maksipituus on ylitetty, päätyy viesti Dead letter -jonoon. Tämänlaisissa tilanteissa on tärkeää saada hälytys, jotta voidaan korjata ongelma ja mahdollisesti siirtää sanoma oikeaan jonoon.

Dead letter -jonovalvontaa varten tehtiin seuraavat toimenpiteet:

- Jonosyvyysvalvontaa varten määriteltiin agentille luotuun *userparameter_activemq.conf* -tiedostoon joustava käyttäjäparametri (flexible userparameter), jolle voitiin antaa parametrina halutun jonon nimi.

- Käyttäjäparametrin komennoksi annettiin agentin suorittama shell-komento, joka käytti tietojen hakemiseen ActiveMQ:n REST API:a. Vastauksena saatu JSON-sanoma parsitaan jq-ohjelmalla ja tuloksena saadaan valvontaobjektin arvo, joka lähetetään Zabbix-palvelimelle.

- Raja-arvoksi määriteltiin, että tilanvaihto tapahtuu, kun edellä määritellyn valvontaobjektin arvo on yli nolla.

- Toimenpiteeksi asetettiin sähköpostin lähettäminen tikettijärjestelmän osoitteeseen.

Sanomien hiljaisuuden valvonta

Hiljaisuuden valvonta on varotoimenpide, jolla tarkkaillaan, kulkeeko viestejä tietyllä tietovirralla ja tietyllä ajanjaksolla. Sen on tarkoitus toimia indikaattorina tilanteessa, jossa voidaan olettaa viestien kulkevan. Hiljaisuuden valvonta aiheuttaa helposti vääriä hälytyksiä (false-positive), jotka voivat aiheuttaa turhaa työtä ja pahimmillaan valvonnan passivoitumista, joten on tärkeää määritellä hälytykset siten, ettei vääriä hälytyksiä tulisi liikaa. Myös tikettijärjestelmän mahdollistamalla automaattisella ratkaisutoiminnolla (autoresolve) voidaan vähentää väärien hälytysten aiheuttamaa kuormitusta, kun tikettijärjestelmä automaattisesti ratkaisee ongelman järjestelmässä.

Integraatioalustalla on määritelty omat sisääntulopisteet tietovirroille, esimerkiksi Yritys Y:n käyttämä SFTP-sisääntulopiste. Näitä hälytysten piiriin laitettavia valvontaobjekteja oli paljon ja manuaalinen luominen käyttöliittymällä olisi ollut raskasta. Zabbixin mallit ovat sisäisesti XML-muotoisia (eXtensible Markup Language) ja näitä on mahdollista exportoida sekä importoida. XSLT:n (Extensible Stylesheet Language Transformations) avulla on mahdollista rakentaa malleja ohjelmallisesti tarjoten halutut parametrit ja luoda geneerisen mallin avulla asiakaskohtaisia malleja. Agentille määriteltiin käyttäjäparametriksi tietokantahaku, joka select-kyselyllä haki tietokannasta sanomien lukumäärän, jotka tietyllä aikavälillä olivat kulkeneet sisääntulopisteen kautta. Mallit ja toimenpiteet luotiin samalla tavalla kuin jonosyvyysvalvonnan tapauksessa.

7. Tulokset ja jatkokehitys

Testausta suoritettiin kehityksen ohessa jatkuvasti. Palvelimien valvontaan saatiin ratkaisulla paljon lisäarvoa, kun tieto palvelimien tilasta voitiin yhdistää integraatio-ongelmiin, esimerkiksi jonosyvyyshälytyksen sattuessa voitiin samalta valvontaratkaisulta tarkastaa jonoa lukevan järjestelmän tila, mikäli se oli valvonnan piirissä. Automaattinen tiketöinti helpotti palvelunhallintaa, kun tiketeille saatiin halutut määreet automaattisesti.

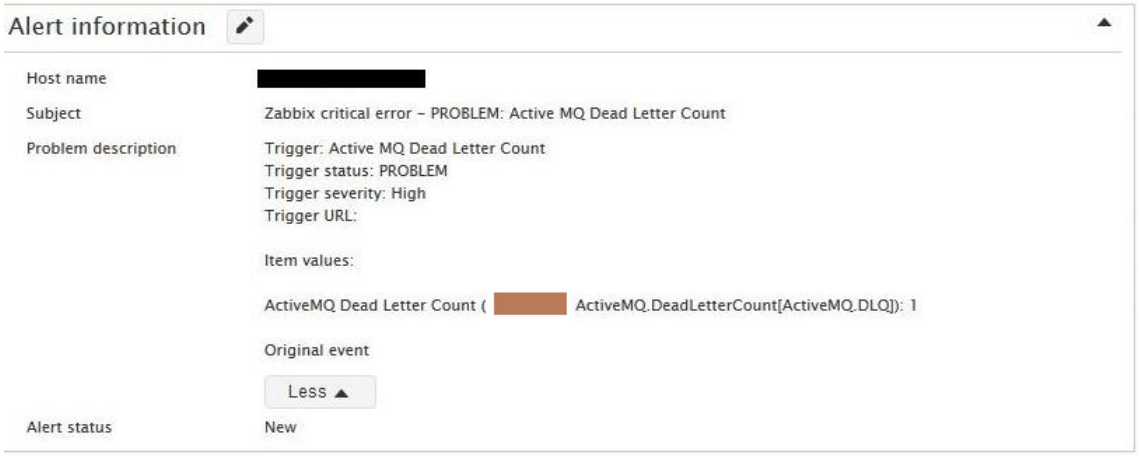
7.1 Onnistumiskriteereiden täyttyminen

Luvussa 4 määriteltiin kahdelle integraatiovalvonnalle onnistumiskriteereitä. Onnistumiskriteerit määriteltiin jonosyvyyksien valvontaa ja hiljaisuuden valvontaa varten. Tässä luvussa esitellään esimerkkivalvontojen testaustavat ja pohditaan asetettujen onnistumiskriteereiden täyttymistä.

Jonosyvyyksien valvontaa testattiin ActiveMQ:n selainkäyttöliittymän kautta laittamalla Dead Letter -jonoon sanoma ja seuraamalla hälytysjärjestelmän toimintaa. Raja-arvon ylitys tuli näkymään käyttöliittymällä sekuntien kuluessa sanoman laittamisesta (kuva 9).

Kuva 9 Jonosyvyyshälytys käyttöliittymällä

Kuvassa 9 näkyy Zabbixin selainkäyttöliittymän etusivulta ”Last 20 issues” -paneelista otettu kuvankaappaus. Ack-kohdasta näkee, onko häiriö huomioitu, ja Actions-kohdasta näkee tehdyt toimenpiteet, kun tekstin päälle vie kursorin. Toimenpiteenä oli sähköpostin lähetys tikettijärjestelmän osoitteeseen, josta tikettijärjestelmä tekee tiketin (kuva 10).



Kuva 10 Jonosyvyyshälytys tikettijärjestelmässä

Kuvassa 10 näkyy osa tikettijärjestelmän alert-tyyppisestä ilmoituksesta, joka sisältää tiedot kohteesta, ongelmasta ja kriittisyydestä. Ilmoituksessa on määritelty myös muun muassa käytetty SLA ja aikarajat ongelman selvittämiseen.

Hiljaisuuden valvontaa testattiin laittamalla valvonnan kohteeksi jokin tietovirta, jolla tiedettiin, ettei määrättynä aikajaksolla ollut kulkenut sanomia, jolloin saatiin Zabbix-hälytys tikettijärjestelmään. Hiljaisuudenvalvonnan haasteina oli saada oikeat valvontamääritykset, jotta turhia hälytyksiä ei tulisi. Vaarana hätiköinnille on false-negative-hälytysten syntyminen, jossa järjestelmä ei hälytä oikeista virheistä. Hiljaisuudenvalvonnan kehittäminen onkin iteratiivinen prosessi, johon oikeat määreet löytyvät ajan kanssa, hiljalleen valvontaa muokkaamalla. Näiden testien avulla varmistettiin esimerkkivalvonnat onnistuneiksi.

7.2 Jatkokehitys

Valvontaratkaisu on vielä suhteellisen alkutekijöissään, ja valvontaa laajennetaan jatkuvasti. Tässä luvussa esitellään jatkokehitysideoita, joista osa on jo pilotointivaiheessa.

7.2.1 Visualisoinnin parantaminen

Zabbixin käyttöliittymä voi tuntua hieman kankealta raportointia silmällä pitäen. Zabbixin keräämää dataa voidaan hyödyntää myös muissa sovelluksissa. Grafana (kuva 11) on avoimen lähdekoodin visualisointiin erikoistunut ohjelmisto, jolla Zabbixin keräämiä tietoja voidaan muuttaa graafiseen muotoon.

Kuva 11. Grafanan käyttöliittymä [19.]

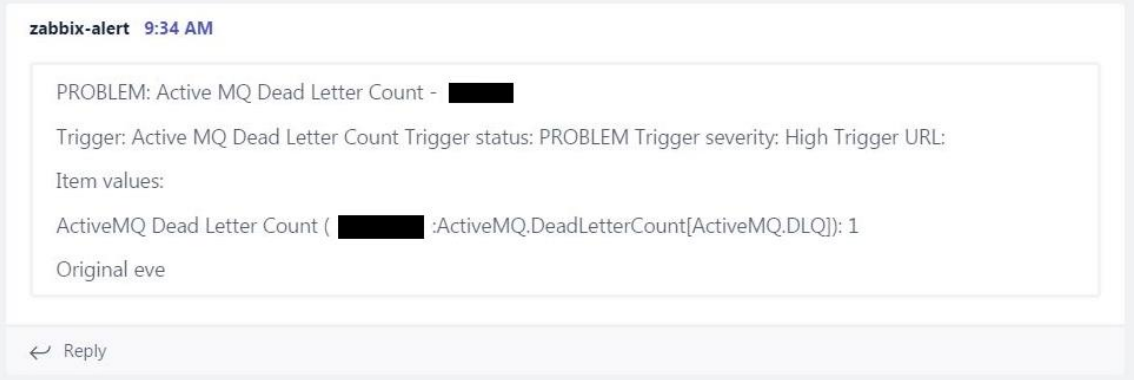
Zabbixista on valmis liitännäinen Grafanaan, joten niiden integrointi on helppoa. Grafana on otettu käyttöön ja sitä käytetään Zabbixin käyttöliittymän ohessa. Tarkoituksena jatkossa on tehdä kaikki visuaalinen esitys Grafanan kautta.

7.2.2 Lokistatistiikka

Loki on tiedosto, johon sovellus tai järjestelmä kerää automaattisesti tietoa tapahtumista. Lokimerkintä on yksittäinen tapahtumatieto, joka voi kertoa esimerkiksi, kuka järjestelmää on käyttänyt tai kertoa tietoa virheestä. Lokienhallintaohjelmiston tarkoituksena on kerätä lokien sisältämiä lokimerkintöjä yhteen ja tarjota helpon tavan etsiä lokitietoa. Zabbixin toiminnallisuuteen ei lähtökohtaisesti kuulu lokienhallinta. Lokienhallintaa käsitteleviä ohjelmistoja on paljon erilaisia. Eräs mietitty vaihtoehto on ELK stack tai ElasticStack, joka on kokoelma avoimen lähdekoodin työkaluista lokivalvontaa varten. Se koostuu kolmesta komponentista, Elasticsearch-hakumoottorista, Logstash-lokienkäsittelyohjelmasta ja Kibana-visualisointityökalusta. Lokienhallinan avulla saataisiin tarkempaa tietoa sovellusten ja käyttöjärjestelmien toiminnasta.

7.2.3 Raportoinnin kehittäminen

Hälytysten saamista kokeiltiin myös MS Teams -ohjelmistoon mukautettujen hälytyskomentojen (custom alertscript) avulla. Mukautetut hälytyskomennot ovat itse luotuja mediatyyppejä Zabbix-palvelimelle, joita voi käyttää natiivien hälytystapojen, kuten sähköpostin tai sms-viestin, sijasta. Zabbix-palvelimelle on kirjoitettu shell-skripti, joka webhook-tiedonvälitysmenetelmällä tekee http POST-kutsuja Teams-palvelimelle. Näin hälytyksille tehtyyn Teams-kanavalle tulee virheistä ilmoitus (kuva 12).



Kuva 12 Jonosyvyyshälytys Teams-ohjelmistolla

Kuvassa on esitelty aikaisempi Dead Letter -jonosta saatu hälytys Teams-ohjelmistolla. Vastaavaa toimintoa voidaan käyttää myös muissa keskusteluohjelmissa, kuten Slack-sovelluksessa.

7.2.4 Hajautettu valvontaratkaisu

Valvonnan laajentuessa Zabbix-komponentit voidaan siirtää eri palvelimille. Voidaan myös käyttää hyväkseen Zabbix-solmuja (node), jolla voidaan luoda hajautettu valvontaratkaisu. Zabbix-solmu toimii itsenäisenä Zabbix-palvelimena, joten se vähentää pääpalvelimen kuormitusta.

7.2.5 Trendien tunnistus

Zabbixissa on versiosta 3.0 lähtien toiminnallisuus trendien tunnistamiseen. Tämän avulla voidaan automaattisesti ennustaa esimerkiksi levytilan loppumisen ajankohta tai huomioida erilaiset normaalista poikkeavat tilanteet.

8. Yhteenveto

Työn tarkoituksena oli tehostaa järjestelmäintegraatioiden valvontaa. Valvontaratkaisulla oletettiin saavutettavan nopeampaa reagointiaikaa integraatioissa tapahtuvien virhetilanteiden tunnistamiseksi, parempaa mahdollisuutta ennaltaehkäistä ongelmien muodostuminen sekä saada nopeammin lisätietoja ongelmien syistä ja seurauksista.

Sekä Zabbix-ohjelmisto että järjestelmäintegraatiot ovat laajoja asioita, joista voisi kertoa vaikka kuinka paljon. Opinnäytetyössä oli kuitenkin pyritty rajaamaan aiheita ja antamaan vain kokonaiskuva aiheiden käsitteistä. Zabbix tarjoaa laajasti myös monia muita ominaisuuksia verkon valvontaan, joita ei tässä työssä käyty läpi, sillä ne eivät olleet oleellisia tai ajankohtaisia toteutettavan integraatiovalvonnan kannalta. Myös järjestelmäintegraatioiden teknologiat tarkemmalla tasolla jätettiin lopputyöstä pois.

Työtä varten oli valittu eräälle ympäristölle kaksi oleellista valvontakomponenttia, joiden onnistumiskriteereitä tutkittiin. Onnistumiskriteerit pyrittiin tekemään sellaisiksi, että niiden onnistuminen tai epäonnistuminen pystyttäisiin varmistamaan täsmällisesti. Työssä mainittuihin onnistumiskriteereihin päästiin työn aikana. Työn ohessa toteutettiin myös monia muita integraatiovalvontoja ja niitä tukevia verkkoinfrastruktuurivalvontoja. Työssä mietittiin myös jatkokehitysideoita valvonnalle, joista osan toteutus on jo tiimissä aloitettu.

Valvontaratkaisu on osoittautunut hyödylliseksi välineeksi ja nopeuttanut virhetilanteiden käsittelyä. Palvelimien ja sovellusten tilasta saatava tieto on kriittistä integraatiopalveluiden kannalta, ja Zabbix oli erinomainen vaihtoehto niiden valvontaan. Valvontaratkaisu on tällä hetkellä tiimin käytössä ja jatkuvan kehityksen alla.

Aihe oli itselleni työn aloitushetkellä luonnollinen, sillä työskentelin tiimissä, jossa ohjaajani kanssa aloimme ottaa valvontaratkaisua käyttöön. Työssä pääsi opettelemaan ja soveltamaan integraatio- ja verkko-osaamista. Opinnäytetyön haasteita oli mm. aiheiden rajaaminen, jotta saisi vain oleelliset asiat mukaan siten, että lukijalle muodostuisi kuitenkin tarpeeksi vahva kokonaiskuva työssä käsitellyistä asioista.

Lähteet

1 Tähtinen S. 2005. Järjestelmäintegraatio. Tarve, Vaihtoehdot, Toteutus. Helsinki: Talentum.

2 Juric M.B., Loganathan, R. & Sarang, P.G. 2007. SOA Approach to Integration. 1st edition. Birmingham: Packt Publishing.

3 Manouvrier B., Ménard L. 2009. Application Integration: EAI, B2B, BPM and SOA. Hoboken: John Wiley & Sons, Inc.

4 Integraatiot. Verkkoaineisto. <https://www.itewiki.fi/opas/integraatiot/>. Luettu 12.10.2017.

5 Pan, A. 2001. Enterprise application integration -- message broker style. Verkkoaineisto <https://www.itworld.com/article/2795762/enterprise-software/enterprise-application-integration----message-broker-style.html>. Luettu 10.10.2017.

6 Krafzig, D., Banke, K. & Slama D. 2004. Enterprise SOA. Service-Oriented Architecture: Best Practices. Verkkokirja. Indianapolis:Prentice Hall. Saatavilla s.1-45.

7 Curry, E. 2008. Message-oriented Middelware. Verkkoaineisto. <https://pdfs.semanticscholar.org/98e1/90fd2ed9e15514f7f5d5ea3dbd8aeb382a9c.pdf>. Luettu 19.11.2017.

8 Message Oriented Middleware and MQSeries. Verkkosaineisto. <http://www.mqseries.net/files/presentations/MQOverview.pdf>. Luettu 20.11.2017

9 Vähimaa, A. 2017. Esb on kuollut – eläköön uusi esb! Verkkoaineisto. <http://www.tivi.fi/Kaikki_uutiset/esb-on-kuollut-elakoon-uusi-esb-6671493>. Luettu 20.11.2017.

10 Rinne, V. 2016. Is it finally time to give up on ESBs? Verkkoaineisto. <https://www.linkedin.com/pulse/finally-time-give-up-esbs-ville-rinne>. Luettu 20.11.2017.

11 Fowler, M. 2014. Microservices. Verkkoaineisto. <https://martinfowler.com/articles/microservices.html>. Luettu 20.11.2017.

12 Niemistö, T. 2017. Onko ESB todella kuollut? Ei vielä... Verkkoaineisto. <http://www.tivi.fi/Kumppaniblogit/digia/onko-esb-todella-kuollut-ei-viela-6683140>. Luettu 20.11.2017.

13 Tien, K. 2016. Microservices versus ESB. verkkoaineisto. <https://blogs.mulesoft.com/dev/microservices-dev/microservices-versus-esb/>. Luettu 20.11.2017.

14 Olups, R. 2010. Zabbix 1.8 Network Monitoring. 1st edition. Birmingham: Packt Publishing.

15 True Open source. Verkkoaineisto. <http://www.zabbix.com/true_open_source>. Luettu 14.10.2017.

16 Vacche, A.D., Lee, S. 2103. Mastering Zabbix. 1st edition. Birmingham: Packt Publishing.

17 Supported platforms. Verkkoaineisto. <https://www.zabbix.com/requirements>. Luettu 14.10.2017.

18 Zabbix General Brochure 3.2. Verkkoaineisto. <https://www.zabbix.com/files/Brochures/General_Brochure_3.2.pdf>. Luettu 14.10.2017.

19 Zobnin, A. 2017. Zabbix plugin for Grafana. Verkkoaineisto. <https://grafana.com/plugins/alexanderzobnin-zabbix-app>. Luettu 20.11.2017.

Liite 1. Jonovalvonnan määritykset

Dead letter -jonovalvonta ActiveMQ:lle.

Tiedostosta */etc/zabbix/zabbix_agentd.d/userparameter_activemq.conf* löytyy ActiveMQ:n käyttäjäparametrit. Alla esimerkki käyttäjäparametrin muodosta Dead Letter -valvontaan.

```
UserParameter=ActiveMQ.DeadLetterCount[*],wget -q --user=**** --
password=**** -O- http://localhost:8161/api/jolo-
kia/read/org.apache.activemq:type=Broker,brokerName=lo-
calhost,destinationType=Queue,destinationName=$1/QueueSize | jq
.'value'| grep --invert-match null  || echo 999999
```

1. määritellään joustava käyttäjäparametri (flexible userparameter), joka hakee jonosyvyyden ActiveMQ:n REST API:a hyödyntäen.
   [*] tarkoittaa, että itemin konfiguroinnissa hyväksytään parametrien asetus, tässä tilanteessa parametrina on jonon nimi.
2. jq komentorivipohjainen JSON käsittelijä, jolla haetaan avaimen “value” arvo.
3. grep –invert-match null hylkää null tulostukset
4. || echo 99999, tulostetaan 99999, mikäli tulostus olisi muuten tyhjä

Raja-arvot määritellään Trigger-objekteille. Alla esimerkki Dead letter -jonolle asetetusta raja-arvosta, jossa tarkastellaan arvoa 10 sekuntia sitten. Jos se on yli 0, niin käsittely siirtyy tapahtumakäsittelijälle.

```
{palvelimenNimi:ActiveMQ.DeadLetterCount[{$AMQ_DEAD_LET-
TER}].last(,10)}<>0
```